

# Efficient Minimization of Higher Order Submodular Functions using Monotonic Boolean Functions

Srikumar Ramalingam<sup>1</sup> Chris Russell<sup>2</sup> Lubor Ladický<sup>3</sup> Philip H.S. Torr<sup>4</sup>

<sup>1</sup>Mitsubishi Electric Research Laboratories, Cambridge, USA,

<sup>2</sup>Queen Mary, University of London, UK,

<sup>3</sup>Oxford University, Oxford, UK,

<sup>4</sup>Oxford Brookes University, Oxford, UK,

**Abstract.** Submodular function minimization is a key problem in a wide variety of applications in machine learning, economics, game theory, computer vision and many others. The general solver has a complexity of  $O(n^6 + n^5 L)$  where  $L$  is the time required to evaluate the function and  $n$  is the number of variables [22]. On the other hand, many useful applications in computer vision and machine learning applications are defined over a special subclasses of submodular functions in which that can be written as the sum of many submodular cost functions defined over cliques containing few variables. In such functions, the pseudo-Boolean (or polynomial) representation [2] of these subclasses are of degree (or order, or clique size)  $k$  where  $k \ll n$ . In this work, we develop efficient algorithms for the minimization of this useful subclass of submodular functions. To do this, we define novel mapping that transform submodular functions of order  $k$  into quadratic ones, which can be efficiently minimized in  $O(n^3)$  time using a max-flow algorithm. The underlying idea is to use auxiliary variables to model the higher order terms and the transformation is found using a carefully constructed linear program. In particular, we model the auxiliary variables as monotonic Boolean functions, allowing us to obtain a compact transformation using as few auxiliary variables as possible. Specifically, we show that our approach for fourth order function requires only 2 auxiliary variables in contrast to 30 or more variables used in existing approaches. In the general case, we give an upper bound for the number of auxiliary variables required to transform a function of order  $k$  using Dedekind number, which is substantially lower than the existing bound of  $2^{2^k}$ .

**Keywords:** submodular functions, quadratic pseudo-Boolean functions, monotonic Boolean functions, Dedekind number, max-flow/mincut algorithm

## 1 Introduction

Many optimization problems in several domains such as operations research, computer vision, machine learning, and computational biology involve *submodular* function minimization. Submodular functions (See Definition 1) are discrete analogues of convex functions [20]. Examples of such functions include cut capacity functions, matroid rank functions and entropy functions. Submodular function minimization techniques may be broadly classified into two categories: efficient algorithms for general submodular functions and more efficient algorithms for subclasses of submodular functions. This paper falls under the second category.

*General solvers:* The role of submodular functions in optimization was first discovered by Edmonds when he gave several important results on the related poly-matroids [4]. Grötschel, Lovász and Schrijver first gave a polynomial-time algorithm for minimization of submodular function using ellipsoid method [7].

Recently several combinatoric and strongly polynomial algorithms [5, 11, 12, 27] have been developed based on the work of Cunningham [3]. The current best strongly polynomial algorithm for minimizing general submodular functions [22] has a run-time complexity of  $O(n^5 L + n^6)$ , where  $L$  is the time taken to evaluate the function and  $n$  is the number of variables. Weakly polynomial time algorithms with a smaller dependence on  $n$  also exist. For example, to minimize the submodular function  $f(x)$  the scaling algorithm of Iwata [13] has a run-time complexity of  $O(n^4 L + n^5 \log M)$ . As before,  $L$  refers to the time required to compute the function  $f$  and  $M$  refers to the maximum absolute value of the function  $f$ .

*Specialized solvers:* There has been much recent interest in the use of higher order submodular functions for better modeling of computer vision and machine learning problems [15, 19, 10]. Such problems typical involve millions of pixels making the use of general solvers highly infeasible. Further, each pixel may take multiple discrete values and the conversion of such a problem to a Boolean one introduces further variables. On the other hand, the cost functions for many such optimization algorithms belong to a small subclass of submodular functions. The goal of this paper is to provide an efficient approach for minimizing these subclasses of submodular functions using a max-flow algorithm.

**Definition 1.** *Submodular functions map  $f : \mathbb{B}^V \rightarrow \mathbb{R}$  and satisfy the following condition:*

$$f(X) + f(Y) \geq f(X \vee Y) + f(X \wedge Y) \quad (1)$$

where  $X$  and  $Y$  are elements of  $\mathbb{B}^n$

In this paper, we use a pseudo-Boolean polynomial representation for denoting submodular functions.

**Definition 2.** *Pseudo-Boolean functions (PBF) take a Boolean vector as argument and return a real number, i.e.  $f : \mathbb{B}^n \rightarrow \mathbb{R}$  [2]. These can be uniquely expressed as multi-linear polynomials i.e. for all  $f$  there exists a unique set of real numbers  $\{a_S : S \in \mathbb{B}^N\}$  :*

$$f(x_1, \dots, x_n) = \sum_{S \subseteq V} a_S \left( \prod_{j \in S} x_j \right), a_S \in \mathbb{R}, \quad (2)$$

where  $a_\emptyset$  is said to be the constant term.

The term *order* refers to the maximum degree of the polynomial. A submodular function of second order involving Boolean variables can be easily represented using a graph such that the minimum cut, computed using a max-flow algorithm, also efficiently minimizes the function. However, max-flow algorithms can not exactly minimize non-submodular functions or some submodular ones of an order greater than 3 [30]. There is a long history of research in solving subclasses of submodular functions both exactly and efficiently using max-flow algorithms [1, 16, 8, 29, 24]. In this paper we propose a novel linear programming formulation that is capable of definitively answering this question: given any pseudo Boolean function, it can derive a quadratic submodular formulation of the same cost, should one exist, suitable for solving with graph-cuts. Where such a quadratic submodular formulation does not exist, it will find the *closest* quadratic submodular function.

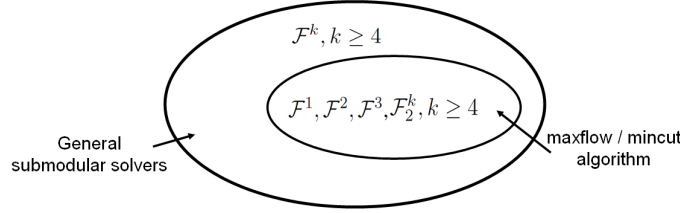
Let  $\mathcal{F}^k$  denote the class of submodular Boolean functions of order  $k$ . It was first shown in [8] that any function in  $\mathcal{F}^2$  can be minimized exactly using a max-flow algorithm. In [1, 16], showed that any function in  $\mathcal{F}^3$  can be transformed into functions in  $\mathcal{F}^2$  and thereby minimized efficiently using max-flow algorithms. The underlying idea is to transform the third order function to a function in  $\mathcal{F}^2$  using extra variables, which we refer to as *auxiliary variables* (AV). In the course of this paper, you will see that these AVs are often

more difficult to handle than variables in the original function and our algorithms are driven by the quest to understand the role of these auxiliary variables and to eliminate the unnecessary ones.

Recently, Zivny et al. made substantial progress in characterizing the class of functions that can be transformed to  $\mathcal{F}^2$ . Their most notable result is to show that not all functions in  $\mathcal{F}^4$  can be transformed to a function in  $\mathcal{F}^2$ . This result stands in strong contrast to the third order case that was positively resolved more than two decades earlier [1]. Using Theorem 5.2 from [23] it is possible to decompose a given submodular function in  $\mathcal{F}^4$  into 10 different groups  $\mathcal{G}_i, i = \{1..10\}$  where each  $\mathcal{G}_i$  is shown in Table 1. Zivny et al. showed that one of these groups can not be expressed using any function in  $\mathcal{F}^2$  employing any number of AVs. Most of these results were obtained by mapping the problem of minimizing submodular functions to a valued constraint satisfaction problem.

### 1.1 Problem Statement and main contributions

*Largest subclass of submodular functions* We are interested in transforming a given function in  $\mathcal{F}^k$  into a function in  $\mathcal{F}^2$  using AVs. As such a transformation is not possible for all submodular functions of order four or more [30], our goal is to implicitly map the largest subclass  $\mathcal{F}_2^k$  that can be transformed into  $\mathcal{F}^2$ . This distinction between the two classes  $\mathcal{F}_2^k$  and  $\mathcal{F}^k$  will be crucial in the remainder of the paper (see Figure 1).



**Fig. 1.** All the function in the classes  $\mathcal{F}^1, \mathcal{F}^2, \mathcal{F}^3$  and  $\mathcal{F}_2^k, k \geq 2$  can be transformed to functions in  $\mathcal{F}^2$  and minimized using the maxflow/mincut algorithm.

**Definition 3.** The class  $\mathcal{F}_2^k$  is the largest subclass of  $\mathcal{F}^k$  such that every function  $f(\mathbf{x}) \in \mathcal{F}_2^k$  has an equivalent quadratic function  $h(\mathbf{x}, \mathbf{z}) \in \mathcal{F}^2$  using AVs  $\mathbf{z} = z_1, z_2, \dots, z_m \in \mathbb{B}^m$  satisfying the following condition:

$$f(\mathbf{x}) = \min_{\mathbf{z} \in \mathbb{B}^m} h(\mathbf{x}, \mathbf{z}), \quad \forall \mathbf{x}. \quad (3)$$

In this paper, we are interested in developing an algorithm to transform every function in this class  $\mathcal{F}_2^k$  to a function in  $\mathcal{F}^2$ .

*Efficient transformation of higher order functions:* We propose a principled framework to transform higher order submodular functions to quadratic ones using a combination of monotonic Boolean functions (MBF) and linear programming. This framework provides several advantages. First we show that the state of an AV in a minimum cost labeling is equivalent to an MBF defined over the original variables. This provides an upper bound on the number of AV given by the Dedekind number [17], which is defined as the total number of MBFs over a set of  $n$  binary variables. In the case of fourth order functions, there are 168 such functions. Using the properties of MBFs and the nature of these AVs in our transformation, we prove that these 168 AVs can be replaced by two AVs.

*Minimal use of AVs:* One of our goals is to use a minimum number( $m$ ) of AVs in performing the transformation of (3). Although, given a fixed choice of  $\mathcal{F}^k$ , reducing the value of  $m$  does not change the complexity of the resulting min/cut algorithm asymptotically, it is crucial in several machine learning and computer vision problems. In general, most image based labeling problems involve millions of pixels and in typical problems, the number of fourth order priors is linearly proportional to the number of pixels. Such problems may be infeasible for large values of  $m$ . A recent work shows that the transformation of functions in  $\mathcal{F}_2^4$  using about 30 additional nodes [31]. On the other hand, we show that we can transform the same class of functions using only 2 additional nodes. Note that this reduction is applicable to every fourth order term in the function. A typical vision problem may involve functions having 10000  $\mathcal{F}_2^4$  terms for an image of size  $100 \times 100$ . Under these parameters, our algorithm will use 20000 AVs, whereas the existing approach [31] would use as large as 300000 AVs. In several practical problems, this improvement will make a significant difference in the running time of the algorithm.

## 1.2 Limitations of Current Approaches and Open Problems

*Decomposition of submodular functions:* Many existing algorithms for transforming higher order functions target the minimization of a single  $k$ -variable  $k^{\text{th}}$  order function. However, the transformation framework is incomplete without showing that a given  $n$ -variable submodular function of  $k^{\text{th}}$  order can be decomposed into several individual  $k$ -variable  $k^{\text{th}}$  order sub-functions. Billionet proved that it is possible to decompose a function in  $\mathcal{F}^3$  involving several variables into 3-variable functions in  $\mathcal{F}^3$  [1]. To the best of our knowledge, the decomposition of fourth or higher order functions is still an open problem. We believe that this problem will be to resolve as, in general, determining if a fourth order function is submodular is co-NP complete [6]. Given this, it is likely that specialized solvers based on max-flow algorithms may never solve the general class of submodular functions. However, this decomposition problem is not a critical issue in machine learning and vision problems. This is because the higher order priors from natural statistics already occur in different sub-functions of  $k$  nodes - in other words, the decomposition is known a priori. This paper only focuses on the transformation of a single  $k$ -variable function in  $\mathcal{F}^k$ . As mentioned above, the solution to this problem is still sufficient to solve large functions with hundreds of nodes and higher order priors in machine learning and vision applications.

*Non-Boolean problems:* The results in this paper are applicable only to set or pseudo-Boolean functions. Many real world problems involve variables that can take multiple discrete values. It is possible to convert any submodular multi-label second order function to their corresponding QBF [9, 26]. One can also transform any multi-labeled higher order function (both submodular and non-submodular) to their corresponding QBF by encoding each multi-label variable using several Boolean variables [25].

*Excess AVs:* The complexity of an efficient max-flow algorithm is  $O((n + m)^3)$  where  $n$  is the number of variables in the original higher order function and  $m$  is the number of AVs. Typically in imaging problems, the number of higher order terms is of  $O(n)$  and the order  $k$  is less than 10. Thus the minimization of the function corresponding to an entire image with  $O(n)$  higher order terms will still have a complexity of  $O((n + n)^3)$ . However when  $m$  becomes at least quadratic in  $n$ , for example, if a higher-order term is defined over every triple of variables in  $V$ , the complexity of the max-flow algorithm will exceed that of a general solver being  $O((n + n^3)^3)$ . Thus in applications involving a very large number of higher order terms, a general solver may be more appropriate.

## 2 Notation and preliminaries

In what follows, we use a vector  $\mathbf{x}$  to denote  $\{x_1, x_2, x_3, \dots, x_n\}$ . Let  $\mathbb{B}$  denote the Boolean set  $\{0, 1\}$  and  $\mathbb{R}$  the set of reals. Let the vector  $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{B}^n$ , and  $\mathbf{V} = \{1, 2, \dots, n\}$  be the set of indices of  $\mathbf{x}$ . Let  $\mathbf{z} = (z_1, z_2, \dots, z_k) \in \mathbb{B}^k$  denote the AVs. We introduce a *set representation* to denote the labellings of  $\mathbf{x}$ . Let  $S_4 = \{1, 2, 3, 4\}$  and let  $\mathcal{P}$  be the power set of  $S_4$ . For example a labeling  $\{x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1\}$  is denoted by the set  $\{1, 3, 4\}$ .

**Definition 4.** The (discrete) derivative of a function  $f(x_1, \dots, x_n)$  with respect to  $x_i$  is given by:

$$\frac{\delta f}{\delta x_i}(x_1, \dots, x_n) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) - f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \quad (4)$$

**Definition 5.** The second discrete derivative of a function  $\Delta_{i,j}(\mathbf{x})$  is given by

$$\begin{aligned} \Delta_{i,j}(\mathbf{x}) &= \frac{\delta}{\delta x_j} \frac{\delta f}{\delta x_i}(x_1, \dots, x_n) \\ &= \left( f(x_1, \dots, x_{i-1}, 1, x_{i+1}, x_{j-1}, 1, x_{j+1}, \dots, x_n) - f(x_1, \dots, x_{i-1}, 0, x_{i+1}, x_{j-1}, 1, x_{j+1}, \dots, x_n) \right) \\ &\quad - \left( f(x_1, \dots, x_{i-1}, 1, x_{i+1}, x_{j-1}, 0, x_{j+1}, \dots, x_n) - f(x_1, \dots, x_{i-1}, 0, x_{i+1}, x_{j-1}, 0, x_{j+1}, \dots, x_n) \right). \end{aligned} \quad (5)$$

Note that it follows from the definition of submodular functions (1), that their second derivative is always non-positive for all  $\mathbf{x}$

## 3 Transforming functions in $\mathcal{F}_2^n$ to $\mathcal{F}^2$

Consider the following submodular function  $f(\mathbf{x}) \in \mathcal{F}_2^n$  represented as a multi-linear polynomial:

$$f(\mathbf{x}) = \sum_{S \in \mathcal{B}^n} a_S \left( \prod_{j \in S} x_j \right), a_S \in \mathbb{R} \quad (6)$$

Let us consider a function  $h(\mathbf{x}, \mathbf{z}) \in \mathcal{F}^2$  where  $\mathbf{z}$  is a set of AVs used to model functions in  $\mathcal{F}_2^n$ . Any general function in  $\mathcal{F}^2$  can be represented as a multi-linear polynomial (consisting of linear and bi-linear terms involving all variables):

$$h(\mathbf{x}, \mathbf{z}) = \sum_i a_i x_i - \sum_{i,j:i>j} a_{i,j} x_i x_j + \sum_l a_l z_l - \sum_{l,m:l>m} a_{l,m} z_l z_m - \sum_{i,l} a_{i,l} x_i z_l \quad (7)$$

The negative signs in front of the bi-linear terms  $(x_i x_j, z_l x_i, z_l z_m)$  emphasize that their coefficients  $(-a_{ij}, -a_{il}, -a_{lm})$  must be non-positive if the function is submodular. We are seeking a function  $h$  such that:

$$f(\mathbf{x}) = \min_{\mathbf{z} \in \mathbb{B}^n} h(\mathbf{x}, \mathbf{z}), \forall \mathbf{x}. \quad (8)$$

Here the function  $f(\mathbf{x})$  is known. We are interested in computing the coefficients  $\mathbf{a}$ , and in determining the number of auxiliary variables required to express a function as a pairwise submodular function. The problem is extremely challenging due to the inherent instability and dependencies within the problem – different choices of parameters cause auxiliary variables to take different states. To explore the space of possible solutions fully, we must characterize what states an AV takes.

### 3.1 Auxiliary Variables as Monotonic Boolean Functions

**Definition 6.** A monotonic (increasing) Boolean function (MBF)  $m : \mathbb{B}^n \rightarrow \mathbb{B}$  takes a Boolean vector as argument and returns a Boolean, s.t if  $y_i \leq x_i, \forall i \implies m(\mathbf{y}) \leq m(\mathbf{x})$

**Lemma 1.** The function  $z_s(\mathbf{x})$  defined as  $\mathbf{x}$  by

$$z_s(\mathbf{x}) = \arg \min_{z_s} \left( \min_{\mathbf{z}'} h(\mathbf{x}, \mathbf{z}', z_s) \right). \quad (9)$$

i.e. that maps from  $\mathbf{x}$  to the Boolean state of  $z_s$  is an MBF (See Definition 6), where  $\mathbf{z}'$  is the set of all auxiliary variables except  $z_s$ .

*Proof.* We consider a current labeling  $\mathbf{x}$  with an induced labeling of  $z_s = z_s(\mathbf{x})$ . We first note

$$h'(\mathbf{x}, z_s) = \min_{\mathbf{z}'} h(\mathbf{x}, \mathbf{z}', z_s) \quad (10)$$

is a submodular function i.e. it satisfies (1). We now consider *increasing* the value of  $\mathbf{x}$ , that is given a current labeling  $\mathbf{x}$  we consider a new labeling  $\mathbf{x}^{(i)}$  such that

$$x_j^{(i)} = \begin{cases} 1 & \text{if } j = i \\ x_j & \text{otherwise.} \end{cases} \quad (11)$$

We wish to prove

$$z_s(\mathbf{x}^{(i)}) \geq z_s(\mathbf{x}) \quad \forall \mathbf{x}, i. \quad (12)$$

Note that if  $z_s(\mathbf{x}) = 0$  or  $x_i = 1$  this result is trivial. This leaves the case:  $z_s(\mathbf{x}) = 1$  and  $x_i = 0$ . It follows from (5) that:

$$\begin{aligned} h'(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, 0) - h'(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, 1) &\geq \\ h'(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, 1) - h'(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, 0). \end{aligned} \quad (13)$$

As, by hypothesis,  $z_s(\mathbf{x}) = 1$  and  $x_i = 0$  we have:

$$h'(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, 0) \geq h'(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, 1). \quad (14)$$

Hence

$$\begin{aligned} h'(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, 0) - h'(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, 0) &\geq \\ h'(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, 1) - h'(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, 0), \end{aligned} \quad (15)$$

and

$$h'(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, 0) \geq h'(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, 1). \quad (16)$$

Therefore  $z_s(\mathbf{x}^{(i)}) = 1$ . Repeated application of the statement gives  $y_i \leq x_i, \forall i \implies z_s(\mathbf{y}) \leq z_s(\mathbf{x})$  as required  $\square$

**Definition 7.** The Dedekind number  $M(n)$  is the number of MBFs of  $n$  variables. Finding a closed-form expression for  $M(n)$  is known as the Dedekind problem [14, 17].

The Dedekind number of known values are shown below:  $M(1) = 3$ , this corresponds to the set of functions:

$$M_1(x_1) \in \{\mathbf{0}, \mathbf{1}, x_1\}, \quad (17)$$

where  $\mathbf{0}$  and  $\mathbf{1}$  are the functions that take any input and return 0 or 1 respectively.  $M(2) = 6$  corresponding to the set of functions:

$$M_2(x_1, x_2) = \{\mathbf{0}, \mathbf{1}, x_1, x_2, x_1 \vee x_2, x_1 \wedge x_2\} \quad (18)$$

Similarly,  $M(3) = 20$ ,  $M(4) = 168$ ,  $M(5) = 7581$ ,  $M(6) \approx 7.8 \times 10^6$ ,  $M(7) \approx 2.4 \times 10^{12}$ , and  $M(8) \approx 5.6 \times 10^{23}$ . For larger values of  $n$ ,  $M(n)$  remains unknown, and the development of a closed form solution remains an active area of research.

**Lemma 2.** *On transforming the largest graph-representable subclass of  $k^{\text{th}}$  order function to pairwise Boolean function, the upper bound on the maximal number of required AVs is given by the Dedekind number  $M(k)$ .*

*Proof.* The proof is straightforward. Consider a general multinomial, of similar form to equation (6), with more than  $M(k)$  AVs. It follows from lemma 1 that at least 2 of the AVs must correspond to the same MBF, and always take the same values. Hence, all references to one of these AV in the pseudo-Boolean representation can be replaced with references to the other, without changing the associated costs. Repeated application of this process will leave us with a solution with at most  $M(k)$  AVs.  $\square$

Although this upper bound is large for even small values of  $k$ , it is much tighter than the existing upper bound of  $S(k) = 2^{2^k}$  (See Proposition 24 in [32]). For even small values of  $k = \{3, \dots, 8\}$  the upper bound using Dedekind's number is much smaller:  $(M(3) = 20, S(3) = 256)(M(4) = 168, S(4) = 65536), (M(5) = 7581, S(5) \approx 4.29 \times 10^9), (M(6) \approx 7.8 \times 10^6, S(6) \approx 1.85 \times 10^{19}), (M(7) \approx 2.4 \times 10^{12}, S(7) \approx 3.4 \times 10^{38})$  and  $(M(8) \approx 5.6 \times 10^{23}, S(8) \approx 1.156 \times 10^{77})$ . Zivny et.al. have emphasized the importance of improving this upper bound. In section 5, we will further tighten the bound for fourth order functions.

Note that this representation of AVs as MBF is over-complete, for example if the MBF of a auxiliary variable  $z_i$  is the constant function  $z_i(\mathbf{x}) = \mathbf{1}$  we can replace  $\min_{\mathbf{z}, z_i} h(\mathbf{x}, \mathbf{z}, z_i)$  with the simpler (i.e. one containing less auxiliary variables) function  $\min_{\mathbf{z}} h(\mathbf{x}, \mathbf{z}, 1)$ . Despite this, this is sufficient preliminary work for our main result:

**Theorem 1.** *Given any function  $f$  in  $\mathcal{F}_2^k$ , the equivalent pairwise form  $f' \in \mathcal{F}^2$  can be found by solving a linear program.*

The construction of the linear program is given in the following section.

## 4 The Linear Program

A sketch of the formulation can be given as follows: In general, the presence of AVs of indeterminate state, given a labeling  $\mathbf{x}$  makes the minimizing an LP non-convex and challenging to solve directly. Instead of optimizing this problem containing AVs of unspecified state, we create an auxiliary variable associated with every MBF. Hence given any labeling  $\mathbf{x}$  the state of every auxiliary variable is fixed a priori, making the problem convex. We show how the constraints that a particular AV must conform to a given MBF can be formulated as linear constraints, and that consequently the problem of finding the closest member of  $f' \in \mathcal{F}^2$  to any pseudo Boolean function is a linear program.

This program will make use of the max-flow linear program formulation to guarantee that the minimum cost labeling of the AVs corresponds to their MBFs. To do this we must first rewrite the cost of equation (7), in a slightly different form. We write:

$$\begin{aligned} f(\mathbf{x}, \mathbf{z}) = & c_\emptyset + \sum_i c_{i,s} (1 - x_i) + \sum_i c_{t,i} x_i + \sum_{i,j:i>j} c_{i,j} x_i (1 - x_j) \\ & + \sum_l c_{l,s} (1 - z_l) + \sum_l c_{t,l} (1 - z_l) + \sum_{l,m:l>m} c_{l,m} z_l (1 - z_m) + \sum_{i,l} c_{i,l} x_i (1 - z_l) \end{aligned} \quad (19)$$

where  $c_\emptyset$  is a constant that may be either positive or negative and all other  $c$  are non-negative values referred to as the *capacity* of an edge. By [16, 1], this form is equivalent to that of (7), in that any function that can be written in form (7), can also be written as (19) and visa versa.

#### 4.1 The Max-flow Linear Program

Under the assumption that  $\mathbf{x}$  is fixed, we are interested in finding a minima of the equation:

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{z}) = & c_\emptyset + \sum_i c_{i,s} (1 - x_i) + \sum_i c_{t,i} x_i + \sum_{i,j:i>j} c_{i,j} x_i (1 - x_j) \\ & + \sum_l c_{l,s} (1 - z_l) + \sum_l c_{t,l} (1 - z_l) + \sum_{l,m:l>m} c_{l,m} z_l (1 - z_m) + \sum_{i,l} c_{i,l} x_i (1 - z_l) \\ = & d_{\mathbf{x},\emptyset} + \sum_l d_{\mathbf{x},l,s} (1 - z_l) + \sum_l d_{\mathbf{x},t,l} (1 - z_l) + \sum_{l,m:l>m} d_{\mathbf{x},l,m} z_l (1 - z_m) \end{aligned} \quad (20)$$

where

$$d_{\mathbf{x},\emptyset} = c_\emptyset + \sum_{i:x_i=0} c_{i,s} + \sum_{i:x_i=1} c_{t,i} + \sum_{i,j:i>j \wedge x_i=1 \wedge x_j=0} c_{i,j} \quad (21)$$

$$d_{\mathbf{x},s,l} = c_{s,l} + \sum_{i:x_i=1} c_{i,l}, \quad d_{\mathbf{x},l,t} = c_{t,l} \text{ and } d_{\mathbf{x},l,m} = c_{l,m}. \quad (22)$$

Then the minimum cost of equation (19) may be found by solving its dual max-flow program. Writing  $\nabla_{\mathbf{x},s}$  for flow from sink, and  $\nabla_{\mathbf{x},t}$  for flow to the sink, we seek

$$\max \nabla_{\mathbf{x},s} + d_{\mathbf{x},\emptyset} \quad (23)$$

Subject to the constraints that

$$\begin{aligned} & f_{\mathbf{x},ij} - d_{\mathbf{x},ij} \leq 0 \quad \forall (i,j) \in E \\ & \sum_{j:(j,i) \in E} f_{\mathbf{x},ji} - \sum_{j:(i,j) \in E} f_{\mathbf{x},ij} \leq 0 \quad \forall i \neq s, t \\ & \nabla_{\mathbf{x},s} + \sum_{j:(j,s) \in E} f_{\mathbf{x},js} - \sum_{j:(s,j) \in E} f_{\mathbf{x},sj} \leq 0 \\ & \nabla_{\mathbf{x},t} + \sum_{j:(j,t) \in E} f_{\mathbf{x},jt} - \sum_{j:(t,j) \in E} f_{\mathbf{x},tj} \leq 0 \\ & f_{\mathbf{x},ij} \geq 0 \quad (i,j) \in E \end{aligned} \quad (24)$$

where  $E$  is the set of all ordered pairs  $(l, m) : \forall l > m$ ,  $(s, l) : \forall l$  and  $(l, t) : \forall t$ , and  $f_{\mathbf{x},i,j}$  corresponds to the flow through the edge  $(i, j)$ .

We will not use this exact LP formulation, but instead rely on the fact that  $f_{\mathbf{x}}(\mathbf{z})$  is a *minimal cost labeling* if and only if there exists a flow satisfying constraints (24) such that

$$f_{\mathbf{x}}(\mathbf{z}) - \nabla_{\mathbf{x},s} - d_{\mathbf{x},\emptyset} \leq 0. \quad (25)$$



## 4.2 Choice of MBF as a set of linear constraints

We are seeking minima of a quadratic pseudo Boolean function of the form (19), where  $\mathbf{x}$  is the variables we are interested in minimizing and  $\mathbf{z}$  the auxiliary variables. As previously mentioned, formulations that allow the state of the auxiliary variable to vary tend to result in non-convex optimization problems. To avoid such difficulties, we specify as the location of minima of  $\mathbf{z}$  as a set hard constraints. We want that:

$$\min_{\mathbf{z}} f_{\mathbf{x}}(\mathbf{z}) = f_{\mathbf{x}}([m_1(\mathbf{x}), m_2(\mathbf{x}), \dots, m_{M(k)}(\mathbf{x})]) \quad \forall \mathbf{x}. \quad (26)$$

where  $f_{\mathbf{x}}$  is defined as in (20), and  $m_1, \dots, m_{M(k)}$  are the set of all possible MBFs defined over  $\mathbf{x}$ . By setting all of the capacities  $d_{i,j}$  to 0, it can be seen that a solution satisfying (26) must exist. It follows from the reduction described in lemma 1, and that all functions that can be expressed in a pairwise form can also be expressed in a form that satisfies these restrictions.

We enforce condition (26) by the set of linear constraints (24) and (25) for all possible choice of  $\mathbf{x}$ . formally we enforce the condition

$$f_{\mathbf{x}}([m_1(\mathbf{x}), \dots, m_{M(k)}(\mathbf{x})]) - \nabla_{\mathbf{x},s} - d_{\mathbf{x},\emptyset} \leq 0. \quad (27)$$

Substituting in (20) we have  $2^k$  sets of conditions, namely,

$$\sum_l d_{\mathbf{x},l,s} (1 - m_l(\mathbf{x})) + \sum_l d_{\mathbf{x},t,l} (1 - m_l(\mathbf{x})) + \sum_{l,m:l>m} d_{\mathbf{x},l,m} m_l(\mathbf{x}) (1 - m_m(\mathbf{x})) - \nabla_{\mathbf{x},s} \leq 0, \quad (28)$$

subject to the set of constraints (24) for all  $\mathbf{x}$ . Note that we make use of the max-flow formulation, and not the more obvious min-cut formulation, as this remains a linear program even if we allow the capacity of edges  $\mathbf{d}^1$  to vary.

*Submodularity Constraints* We further require that the quadratic function is submodular or equivalently, the capacity of all edges  $c_{i,j}$  is non-negative. This can be enforced by the set of linear constraints that

$$c_{i,j} \geq 0 \quad \forall i, j. \quad (29)$$

## 4.3 Finding the nearest submodular Quadratic Function

We now assume that we have been given an arbitrary function  $g(\mathbf{x})$  to minimize, that may or may not lie in  $\mathcal{F}^k$ . We are interested in finding the closest possible function in  $\mathcal{F}^2$  to it. To find the closest function to it (under the  $L_1$  norm), we minimize:

---

<sup>1</sup> In itself  $\mathbf{d}$  is just a notational convenience, being a sum of coefficients in  $\mathbf{c}$ .

$$\min_{\mathbf{c}} \sum_{\mathbf{x} \in \mathbb{B}^k} \left| g(\mathbf{x}) - \min_{\mathbf{z}} f(\mathbf{x}, \mathbf{z}) \right| = \quad (30)$$

$$\min_{\mathbf{c}} \sum_{\mathbf{x} \in \mathbb{B}^k} \left| g(\mathbf{x}) - f(\mathbf{x}, \mathbf{m}(\mathbf{x})) \right| = \quad (31)$$

$$\begin{aligned} \min_{\mathbf{c}} \sum_{\mathbf{x} \in \mathbb{B}^k} \left| g(\mathbf{x}) - \left( c_{\emptyset} + \sum_i c_{i,s} (1 - x_i) + \sum_i c_{t,i} x_i + \sum_{i,j:i>j} c_{i,j} x_i (1 - x_j) \right. \right. \\ \left. \left. + \sum_l c_{l,s} (1 - m_l(\mathbf{x})) + \sum_l c_{t,l} (1 - m_l(\mathbf{x})) + \sum_{l,m:l>m} c_{l,m} m_l(\mathbf{x}) (1 - m_m(\mathbf{x})) \right. \right. \\ \left. \left. + \sum_{i,l} c_{i,l} x_i (1 - m_l(\mathbf{x})) \right) \right| \end{aligned} \quad (32)$$

where  $\mathbf{m}(\mathbf{x}) = [m_1(\mathbf{x}), \dots, m_{M(k)}(\mathbf{x})]$  is the vector of all MBFs over  $\mathbf{x}$ , and subject to the family of constraints set out in the previous subsection. Note that expressions of the form  $\sum_i |g_i|$  can be written as  $\sum_i h_i$  subject to the linear constraints  $h_i > g_i$  and  $h_i > -g_i$  and this is a linear program.  $\square$

#### 4.4 Discussion

Several results follow from this. In particular, if we consider a function  $g$  of the same form as equation (2) the set of equations such that

$$\min_{\mathbf{c}} \sum_{\mathbf{x} \in \mathbb{B}^k} \left| g(\mathbf{x}) - \min_{\mathbf{z}} f(\mathbf{x}, \mathbf{z}) \right| = 0 \quad (33)$$

exactly defines a linear polytope for any choice of  $|\mathbf{x}| = k$ , and this result holds for any choice of basis functions.

Of equal note, the convex-concave procedure [28], is a generic move-making algorithm that finds local optima by successively minimizing a sequence of convex (i.e. tractable) upper-bound functions that are tight at the current location ( $\mathbf{x}'$ ). [21] showed how this could be similarly done for quadratic Boolean functions, by decomposing them into submodular and supermodular components. The work [18] showed that any function could be decomposed into a quadratic submodular function, and an additional overestimated term. Nevertheless, this decomposition was not optimal, and they did not suggest how to find an optimal overestimation. The optimal overestimation which lies in  $\mathcal{F}^2$  for a cost function defined over a clique  $\mathbf{g}$  may be found by solving the above LP subject to the additional requirements:

$$g(\mathbf{x}) \leq f(\mathbf{x}, \mathbf{z}) \quad \forall \mathbf{x} \quad (34)$$

$$g(\mathbf{x}') \geq f(\mathbf{x}', \mathbf{z}) \quad (35)$$

*Efficiency concerns* As we consider larger cliques, it becomes less computationally feasible to use the techniques discussed in this section, at least without pruning the number of auxiliary variables considered. As previously mentioned, constant AVs and AVs that corresponds to that of a single variable in  $x$  i.e.  $z_l = x_i$  can be safely discarded without loss of generality. In the following section, we show that a function in  $\mathcal{F}_2^4$  can be represented by only two AVs, rather than 168 as suggested by the number of possible MBF. However, in the general case a minimal form representation eludes us. As a matter of pragmatism, it may be useful to attempt to solve the LP of the previous section without making use of any AV, and to successively introduce new variables, until a minimum cost solution is found.

## 5 Tighter Bounds: Transforming functions in $\mathcal{F}_2^4$ to $\mathcal{F}^2$

Consider the following submodular function  $f(x_1, x_2, x_3, x_4) \in \mathcal{F}^4$  represented as a multi-linear polynomial:

$$f(x_1, x_2, x_3, x_4) = a_0 + \sum_i a_i x_i + \sum_{i>j} a_{ij} x_i x_j + \sum_{i>j>k} a_{ijk} x_i x_j x_k + a_{1234} x_1 x_2 x_3 x_4, \quad \Delta_{ij}(\mathbf{x}) \leq 0 \quad (36)$$

where  $i, j, k = S_4$  and  $\Delta_{ij}(\mathbf{x})$  is the discrete second derivative of  $f(\mathbf{x})$  with respect to  $x_i$  and  $x_j$ .

Consider a function  $h(x_1, x_2, x_3, x_4, z_s) \in \mathcal{F}^2$  where  $z_s$  is an AV used to model functions in  $\mathcal{F}^4$ . Any general function in  $\mathcal{F}^2$  can be represented as a multi-linear polynomial (consisting of linear and bilinear terms involving all five variables):

$$h(x_1, x_2, x_3, x_4, z_s) = b_0 + \sum_i b_i x_i - \sum_{i>j} b_{ij} x_i x_j - (g_s - \sum_{i=1}^4 g_{s,i} x_i) z_s, \quad b_{ij} \geq 0, g_{s,i} \geq 0, i, j \in S_4. \quad (37)$$

The negative signs in front of the bilinear terms  $(x_i x_j, z_s x_i)$  emphasize that their coefficients  $(-b_{ij}, -g_{s,i})$  must be non-positive to ensure submodularity. We have the following condition from equation (3):

$$f(x_1, x_2, x_3, x_4) = \min_{z_s \in \mathbb{B}} h(x_1, x_2, x_3, x_4, z_s), \forall \mathbf{x}. \quad (38)$$

Here the coefficients  $(a_i, a_{ij}, a_{ijk}, a_{ijkl})$  in the function  $f(\mathbf{x})$  are known. We wish to compute the coefficients  $(b_i, b_{ij}, g_s, g_{s,n})$  where  $i, j \in \mathbf{V}, i \neq j, n \in S_4$ . If we were given  $(g_s, g_{s,i})$  then from equations (37) and (38) we would have

$$z_s = \begin{cases} 1 & \text{if } g_s - \sum_{i=1}^4 g_{s,i} x_i < 0, \\ 0 & \text{otherwise.} \end{cases} \quad (39)$$

The value of  $z_s$  that minimizes equation (38) is dependent both upon the assignment of  $\{x_1, x_2, x_3, x_4\}$  and upon the coefficients  $(g_s, g_{s,1}, g_{s,2}, g_{s,3}, g_{s,4})$ . The four variables  $x_1, x_2, x_3$  and  $x_4$  can be assigned to 16 different labellings of  $(x_1, x_2, x_3, x_4)$  giving 16 equations in the following form:

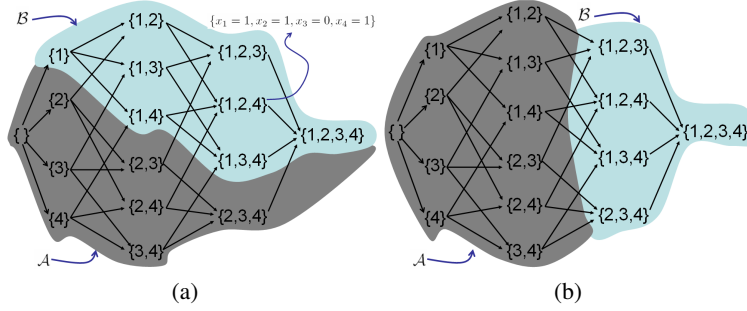
$$f(x_1, x_2, x_3, x_4) = \underbrace{h(x_1, x_2, x_3, x_4, 0)}_{h_1} + \underbrace{\min_{z_s \in \mathbb{B}} (g_s - \sum_{i=1}^4 g_{s,i} x_i) z_s}_{h_2} \quad (40)$$

$\underbrace{\hspace{10em}}_h$

The function  $h_1$  is the part of  $h$  not dependent on  $z_s$ , and  $h_2$  is the part dependent on  $z_s$ . Our main result is to prove that any function  $h \in \mathcal{F}^2$  can be transformed to a function  $h'(x_1, x_2, x_3, x_4, z_{j1}, z_{j2}) \in \mathcal{F}^2$  involving only two auxiliary variables  $z_{j1}$  and  $z_{j2}$ . Using this result we can transform a given function  $f(x_1, x_2, x_3, x_4) \in \mathcal{F}_2^4$ , the form of which we characterize later, to a function  $h'(x_1, x_2, x_3, x_4, z_{j1}, z_{j2}) \in \mathcal{F}^2$ .

Let  $\mathcal{A}$  be the family of sets corresponding to labellings of  $\mathbf{x}$  such that:  $z_s = 0 = \arg \min_{z_s} h(\mathbf{x}, z_s)$ . In the same way let  $\mathcal{B}$  be the family of sets corresponding to labellings of  $\mathbf{x}$  such that:  $z_s = 1 = \arg \min_{z_s} h(\mathbf{x}, z_s)$ . These sets  $\mathcal{A}$  and  $\mathcal{B}$  partition  $\mathbf{x}$ , as defined below:

**Definition 8.** A partition divides  $\mathcal{P}$  into sets  $\mathcal{A}$  and  $\mathcal{B}$  such that  $\mathcal{A} = \{\mathcal{S}(\mathbf{x}) : 0 = \arg \min_{z \in \mathbb{B}} h(\mathbf{x}, z), \mathbf{x} \in \mathbb{B}^4\}$  and  $\mathcal{B} = \mathcal{P} \setminus \mathcal{A}$ . Note that  $\emptyset \in \mathcal{A}$ .



**Fig. 2.** Hasse diagrams sample partitions. Here, we use set representation for denoting the labellings of  $(x_1, x_2, x_3, x_4)$ . For example the set  $\{1, 2, 4\}$  is equivalent to the labelling  $\{x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1\}$ . In (a),  $\mathcal{A} = \{\{\}, \{2\}, \{3\}, \{4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{2, 3, 4\}\}$  and  $\mathcal{B} = \{\{1\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, S_4\}$ . (a) and (b) are examples of partitions. On searching the space of all possible partitions ( $2^{16}$ ) we found that only 168 partitions belong to this class. These are the only partitions which will be useful in our analysis because any arbitrary AV must be associated with one of these 168 partitions. (See text for the relation between these partitions and MBF s).

In the rest of the paper, we say that the AV  $z_s$  is associated with  $[\mathcal{A}, \mathcal{B}]$  or denote it by  $z_s : [\mathcal{A}, \mathcal{B}]$ . We illustrate the concept of a *partition* in figure 2.

From lemma 2, we could use 168 different AVs in our transformation. However, we show that the same class can be represented using only two AVs. In other words, all existing partitions could be converted to these two reference partitions represented by two AVs taking the states shown below.

**Definition 9.** *The forward reference partition  $[\mathcal{A}_f, \mathcal{B}_f]$  takes the form:*

$$B \in \mathcal{B}_f \iff |B| \geq 3, \mathcal{A}_f = \mathcal{P} \setminus \mathcal{B}_f \quad (41)$$

*On the other hand, a backward reference partition  $[\mathcal{A}_b, \mathcal{B}_b]$  is shown below:*

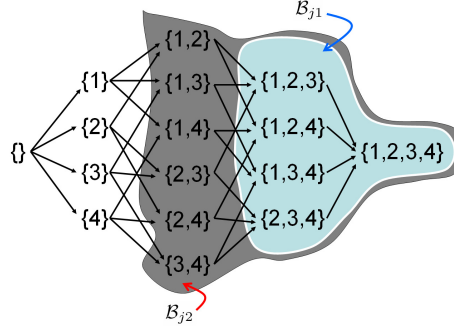
$$B \in \mathcal{B}_b \iff |B| \geq 2, \mathcal{A}_b = \mathcal{P} \setminus \mathcal{B}_b \quad (42)$$

*The forward and backward reference partitions are shown in figure 3. Note that these reference partitions satisfy the properties of a matroid. Here we treat  $\mathcal{A}$  as the family of subsets of the ground set  $S_4$ . More specifically, these reference partitions satisfy the conditions of a uniform matroid (see appendix).*

We approach this problem by first considering the simplified case in which no interactions between AVs are allowed. This is covered in section 5.1, while section 5.2 builds on these results to handle the case of pairwise interactions between AV.

### 5.1 Non-interacting AVs

Here we study the role of AV independently. In other words, we don't consider the interaction of AVs that involve bilinear terms such as  $z_i z_j$ . The following lemmas and theorems enable the replacement of AVs with other AVs closer to the reference partitions. By successively applying replacement algorithms, we gradually replace all the AVs using with the two AVs in forward and backward reference partitions.



**Fig. 3.** The two matroidal generators used to represent all functions in  $\mathcal{F}_2^4$ . Note that the bilinear term  $z_{j1}z_{j2}$  is active, i.e.  $z_{j1}z_{j2} = 1$ , in the region of overlap.

**Lemma 3.** Let  $z_s : [\mathcal{A}_s, \mathcal{B}_s]$  be an AV in a function  $h(\mathbf{x}, z_s)$  in  $\mathcal{F}^2$ , then  $h$  can be transformed to some function  $h'(\mathbf{x}, z_t)$  in  $\mathcal{F}^2$  involving  $z_t : [\mathcal{A}_t, \mathcal{B}_t]$ , such that for all  $B \in \mathcal{B}_t$ ,  $|B| \geq 2$ .

*Proof.* We say that a function  $h$  can be transformed to  $h'$  if  $\min_{z_s} h(\mathbf{x}, z_s) = \min_{z_t} h'(\mathbf{x}, z_t), \forall \mathbf{x}$ . It does not imply that  $h(\mathbf{x}, z_s) = h'(\mathbf{x}, z_t), \forall \mathbf{x}$ . We first consider the case where  $\emptyset \in \mathcal{B}_s$ . If this is the case,  $\arg \min_{z_t} h'(\mathbf{x}, z_t) = 0 \forall \mathbf{x}$ . Hence we can transform  $h(\mathbf{x}, z_s)$  to  $h'(\mathbf{x})$  and the lemma holds trivially. Next we assume that there exists a singleton  $\{e\} \in \mathcal{B}_s$ , i.e.  $\{e\}$  is  $\{1\}, \{2\}, \{3\}$  or  $\{4\}$ . We decompose  $h$  as:

$$\min_{z_s} h(x_1, x_2, x_3, x_4, z_s) = h_1(x_1, x_2, x_3, x_4) + \underbrace{\min_{z_s} (g_s - \sum_{i=1}^4 g_{s,i} x_i) z_s}_{h_2}$$

where  $h_2$  is the part of  $h$  dependent on  $z_s$ .

$$\min_{z_s} h_2 = \min_{z_s} ((g_s - g_{s,e}) x_e z_s + (g_s - g_s x_e - \sum_{i=S_4 \setminus e} g_{s,i} x_i) z_s).$$

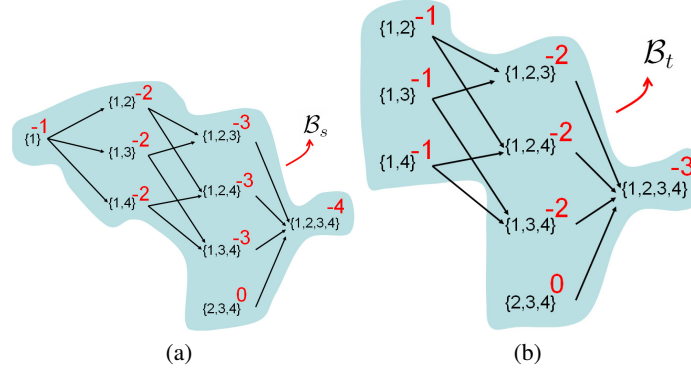
As  $(e) \in \mathcal{B}_s$ ,  $g_s - g_{s,e} \leq 0$ . As a result,  $z_s = 1$  when  $x_e = 1$ , i.e.  $x_e \implies z_s$  or  $x_e z_s = x_e$ . In the above equation we replace  $x_e z_s$  using simply  $x_e$  to obtain the following:

$$\min_{z_s} h_2 = \min_{z_s} ((g_s - g_{s,e}) x_e + (g_s - g_s x_e - \sum_{i=S_4 \setminus e} g_{s,i} x_i) z_s).$$

The decomposition of the original function can then be written, replacing  $z_s$  by  $z_t$ :

$$h' = \underbrace{h_1 + (g_s - g_{s,e}) x_e}_{h'_1} + \underbrace{(g_s - g_s x_e - \sum_{i=S_4 \setminus e} g_{s,i} x_i) z_t}_{h'_2}.$$

A sample reduction for this lemma is shown in figure 4. Note that  $h'_2$  equals 0 for the singleton  $\{e\}$ . Similarly any other singleton  $\{e'\}$  can also be removed from  $\mathcal{B}_s$  using the same approach. After repeated application, our final partition,  $\mathcal{B}_t$  does not contain any singletons.  $\square$



**Fig. 4.** An example of lemma 3. The AV  $z_s$  is replaced by  $z_t$  and the associated partitions  $[\mathcal{A}_s, \mathcal{B}_s]$  and  $[\mathcal{A}_t, \mathcal{B}_t]$  are shown in (a) and (b) respectively. The initial and the final set of parameters are given by:  $(g_s = 3, g_{s,1} = 4, g_{s,2} = 1, g_{s,3} = 1, g_{s,4} = 1), (g_t = 3, g_{t,1} = 3, g_{t,2} = 1, g_{t,3} = 1, g_{t,4} = 1)$ . In the initial partition we have the singleton  $\{1\} \in \mathcal{B}_s$ . After the transformation all the singletons  $\{e\} \in \mathcal{A}_t$ .

**Lemma 4.** Any function  $h(\mathbf{x}, z_s)$  in  $\mathcal{F}^2$  with  $z_s$  associated with the partition  $[\mathcal{A}_s, \mathcal{B}_s]$  satisfying the condition  $\mathcal{B}_s \subseteq \mathcal{B}_f$  can be transformed to some function  $h'(\mathbf{x}, z_f)$  in  $\mathcal{F}^2$  with  $z_f$  belonging to the forward reference partition  $[\mathcal{A}_f, \mathcal{B}_f]$ . The same result holds for backward partition.

*Proof.* The proof is by construction. Let the parameters of the partition  $[\mathcal{A}_s, \mathcal{B}_s]$  be  $(g_s, g_{s,1}, g_{s,2}, g_{s,3}, g_{s,4})$ . Our goal is to compute a new set of parameters  $(g_f, g_{f,1}, g_{f,2}, g_{f,3}, g_{f,4})$  corresponding to the forward reference partition such that the associated functions keep the same value at the minimum:

$$\min_{z_f} h'(\mathbf{x}, z_f) = \min_{z_s} h(\mathbf{x}, z_s), \forall \mathbf{x} \quad (43)$$

$$\min_{z_f} (h'_1(\mathbf{x}) + h'_2(\mathbf{x}, z_f)) = \min_{z_s} (h_1(\mathbf{x}) + h_2(\mathbf{x}, z_s)), \forall \mathbf{x} \quad (44)$$

$$\min_{z_f} (h'_2(\mathbf{x}, z_f)) = \min_{z_s} (h_2(\mathbf{x}, z_s)), \forall \mathbf{x} \quad (45)$$

We can rewrite  $h_2$  and  $h'_2$  using  $\kappa$  function:

$$\min_{z_f} \kappa(f, S) z_f = \min_{z_s} \kappa(s, S) z_s, \forall S \in \mathcal{P} \quad (46)$$

By substituting the values of  $z_s$  and  $z_f$  for all  $S \in \mathcal{P}$  we obtain five equations with five unknowns  $(g_f, g_{f,1}, g_{f,2}, g_{f,3}, g_{f,4})$ . We rewrite the equations as:

$$\underbrace{\begin{pmatrix} 1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 \\ 1 & 0 & -1 & -1 & -1 \\ 1 & -1 & 0 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 \end{pmatrix}}_{\mathcal{H}} \begin{pmatrix} g_f \\ g_{f,1} \\ g_{f,2} \\ g_{f,3} \\ g_{f,4} \end{pmatrix} = \begin{pmatrix} \min(0, \kappa(s, \{2, 3, 4\})) \\ \min(0, \kappa(s, \{1, 3, 4\})) \\ \min(0, \kappa(s, \{1, 2, 4\})) \\ \min(0, \kappa(s, \{1, 2, 3\})) \\ \min(0, \kappa(s, S_4)) \end{pmatrix} \quad (47)$$

The solution to the above linear system is unique because  $\mathcal{H}$  is of rank 5. Now we show that the solution satisfies submodularity condition and corresponds to the forward reference partition. Submodularity is ensured by the constraint that the parameters  $(g_{f,1}, g_{f,2}, g_{f,3}, g_{f,4})$  are all non-negative. Using equation (47) and the non-negativity of original variables  $(g_{s,i})$  we obtain the following:

$$g_{f,i} = \min(0, \kappa(s, S_4 \setminus i)) - \min(0, \kappa(s, S_4)) \quad (48)$$

$$\kappa(s, S_4) \leq \kappa(s, S_4 \setminus i) \quad (49)$$

From these equations we can show that  $g_{f,i}$  is always non-negative:

$$g_{f,i} = \begin{cases} 0 & \text{if } \kappa(s, S_4) \geq 0 \text{ and } \kappa(s, S_4 \setminus i) \geq 0 \\ -\kappa(s, S_4) & \text{if } \kappa(s, S_4) \leq 0 \text{ and } \kappa(s, S_4 \setminus i) = 0 \\ \kappa(s, S_4 \setminus i) - \kappa(s, S_4) & \text{if } \kappa(s, S_4) \leq 0 \text{ and } \kappa(s, S_4 \setminus i) \leq 0 \end{cases} \quad (50)$$

We now prove that the computed parameters correspond to the forward reference partition:

$$S \in \begin{cases} \mathcal{B}_f & \text{if } |S| \geq 3 \\ \mathcal{A}_f & \text{otherwise} \end{cases} \quad (51)$$

From equation (47) it follows that any set  $S$ , such that  $|S| \geq 3$ , exists in  $\mathcal{B}_f$ . We need to prove the remaining case where  $|S| \leq 3$ . To do this, we consider  $S = \{i, j\} = S_4 \setminus \{k, l\}$  and examine its partition coefficients:

$$\begin{aligned} \kappa(f, \{i, j\}) &= \kappa(f, \{i, j, k\}) + g_{f,k} \\ \kappa(f, \{i, j\}) &= \kappa(f, \{i, j, k\}) + ((\kappa(f, \{i, j, l\}) - \kappa(f, \{i, j, k, l\})) \\ \kappa(f, \{i, j\}) &= \min(0, \kappa(s, \{i, j, k\})) + \min(0, \kappa(s, \{i, j, l\})) - \min(0, \kappa(s, \{i, j, k, l\})) \end{aligned}$$

As in table 2 (see appendix),  $\kappa(f, \{i, j\})$  has four possible values and  $\kappa(f, \{i, j\}) \geq 0$  in all. As each set  $S : |S| = 2$  exist in  $\mathcal{A}_f$ , every other set with a cardinality less than two must also exist in  $\mathcal{A}_f$ . Hence, for every partition  $\mathcal{A}_s, \mathcal{B}_s$  satisfying  $\mathcal{B}_s \subseteq \mathcal{B}_f$ , we can compute an equivalent reference partition  $[\mathcal{A}_f, \mathcal{B}_f]$ .  $\square$

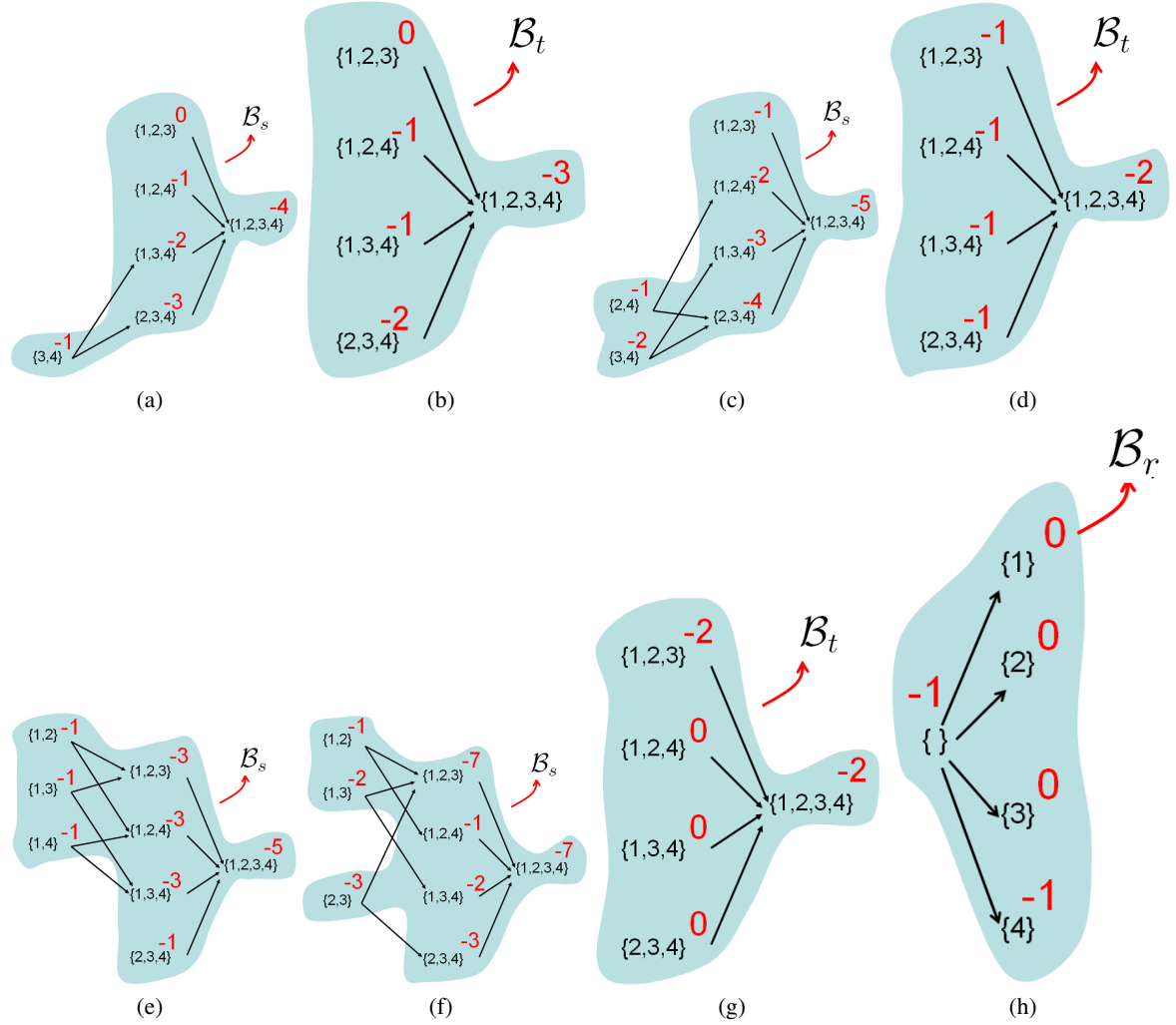
**Lemma 5.** *Let  $P = \{i, j, k, l\} = S_4$  and let  $z_s$  be the auxiliary variable in  $h(\mathbf{x}, z_s)$  associated with the partition  $[\mathcal{A}_s, \mathcal{B}_s]$ . If both  $A$  and  $B = P \setminus A$  are elements of  $\mathcal{B}_s$ , then it is not possible to have both  $C$  and  $D = P \setminus C$  in  $\mathcal{A}_s$ .*

*Proof.* The statement follows by contradiction. Let  $\{A, B\}$ , where  $B = P \setminus A$ , exist in  $\mathcal{B}_s$ . The partition coefficients of  $A$  and  $B$  with respect to  $z_1$  are shown below:

$$\begin{aligned} \kappa(s, A) &= g_s - \sum_{i=1}^4 \mathbf{1}_i^A \leq 0 \\ \kappa(s, B) &= g_s - \sum_{i=1}^4 \mathbf{1}_i^B \leq 0 \end{aligned}$$

Note that  $A \cup B = \{i, j, k, l\}$  and  $A \cap B = \emptyset$ . Hence by summing the above equations we get the following:

$$2g_s - g_{s,i} - g_{s,j} - g_{s,k} - g_{s,l} \leq 0 \quad (52)$$



**Fig. 5.** Examples for the four cases in tables 3, 4, 5 and 6. In the first case the transition in (a) is mapped to that in (b) and the associated parameters are given by:  $((g_s = 6, g_{s,1} = 1, g_{s,2} = 1, g_{s,3} = 1, g_{s,4} = 1), (g_t = 5, g_{t,1} = 1, g_{t,2} = 2, g_{t,3} = 2, g_{t,4} = 3))$ . The generated pairwise term, independent of AVs, is  $-x_3x_4$ . The second case is in (c) and (d) with the parameters  $((5, 1, 2, 3, 4, 5), (2, 1, 1, 1, 1))$  (shown in the same order as the earlier one) and the pairwise function is  $-x_2x_4 - 2x_3x_4$ . The third case is in (e) and (d) with the parameters  $((5, 4, 1, 1, 1), (2, 1, 1, 1, 1))$  along with the pairwise function  $-x_1x_2 - x_1x_3 - x_1x_4$ . The final case is in (f), (g) and (h), as the final function has two AVs  $z_2$  and  $z_3$ . The function consisting of unary and pairwise terms independent of AVs is given by  $1 - x_1 - x_2 - x_3 - x_1x_3 - 2x_2x_3$ . Corresponding parameters are given by:  $((g_s = 8, g_{s,1} = 4, g_{s,2} = 5, g_{s,3} = 6, g_{s,4} = 0), (g_t = 4, g_{t,1} = 2, g_{t,2} = 2, g_{t,3} = 2, g_{t,4} = 0), (g_r = 2, g_{r,1} = 1, g_{r,2} = 1, g_{r,3} = 1, g_{r,4} = 0))$



Assume now that a different pair  $\{C, D\}$ , where  $D = P \setminus C$  exist in  $\mathcal{A}_s$ . By summing their corresponding partition coefficients we get the following equation:

$$2g_s - g_{s,i} - g_{s,j} - g_{s,k} - g_{s,l} \geq 0, \quad (53)$$

Equations 52 and 53 lead to a contradiction, therefore the lemma holds.  $\square$

**Theorem 2.** Any function  $h(\mathbf{x}, z_s)$  in  $\mathcal{F}^2$  with  $z_s$  associated with  $[\mathcal{A}_s, \mathcal{B}_s]$ , such that  $\forall B \in \mathcal{B}_s, |B| \geq 2$ , can be transformed to another function  $h''(\mathbf{x}, z_f, z_b)$  in  $\mathcal{F}^2$  without any  $z_f z_b$  terms, where  $z_f$  and  $z_b$  are AV correspond to the forward and backward reference partitions respectively.

*Proof.* Our proof by construction takes the form of a two-step procedure. In the first stage every function  $h(\mathbf{x}, z_s)$  is transformed to  $h'(\mathbf{x}, z_t, z_r)$  where  $z_t$  and  $z_r$  are associated with the partition  $[\mathcal{A}_t, \mathcal{B}_t]$  and the backward partition  $[\mathcal{A}_r, \mathcal{B}_r]$  respectively and satisfy the conditions  $\mathcal{B}_t \subseteq \mathcal{B}_f$  and  $\mathcal{B}_r \subseteq \mathcal{B}_b$ . In the second step we use lemma 4 to transform  $h'(\mathbf{x}, z_t, z_r)$  to  $h''(\mathbf{x}, z_f, z_b)$ . In most cases only one partition, either the forward or the backward, is used.

$$\min_{z_s} h_2(\mathbf{x}, z_s) = \min_{z_s} \kappa(s, S) z_s, \forall S \in \mathcal{P} \quad (54)$$

$$\min_{z_s} h(\mathbf{x}, z_s) = \sum_{i=1}^4 a_i x_i + \sum_{i=1}^4 \sum_{j, i \neq j}^4 a_{i,j} x_i x_j + \min_{z_t} \kappa(t, S) z_t + \min_{z_r} \kappa(r, S) z_r, \forall S \in \mathcal{P} \quad (55)$$

The key idea is to decompose  $h_2$  into functions of unary and pairwise terms involving only  $\mathbf{x}$  and functions involving new auxiliary variables  $z_t$  and  $z_r$ . Consider the condition  $|B| \geq 2$ . A degenerate case occurs where  $|B| \geq 3$ ; here we can directly use lemma 4 to obtain our desired result. We now consider the cases where at least one set  $S \in \mathcal{B}_s$  has cardinality two and show a transformation similar to the general one of (55). Tables 3, 4, 5 and 6 in the appendix contain details of the decomposition.

After the decomposition the new partitions  $[\mathcal{A}_t, \mathcal{B}_t]$  and  $[\mathcal{A}_r, \mathcal{B}_r]$  satisfy the conditions  $\mathcal{B}_t \subseteq \mathcal{B}_f$  and  $\mathcal{B}_r \subseteq \mathcal{B}_b$ . To show this, we first consider the case where exactly one set  $S \in \mathcal{B}_s$  has a cardinality of 2. There are six such occurrences, and all of them are symmetrical. The transformation for this case is in table 3.

Next, consider the case where exactly two sets of cardinality two exist in  $\mathcal{B}_s$ . Although there are 15  $\binom{6}{2}$  possible cases, they must all be of the form  $\{\{i, j\}, \{k, l\}\}$  or  $\{\{i, j\}, \{j, k\}\}$ . The first sub-case is prohibited because the presence of the mutually exclusive pair  $\{\{i, j\}, \{k, l\}\}$  would not permit any other mutually exclusive pair  $\{\{i, k\}, \{j, l\}\}$  to exist in  $\mathcal{A}_s$  as per lemma 5. The transformation for the latter case is in table 4.

Finally, consider the case where exactly three sets of cardinality two exist in  $\mathcal{B}_s$ . The 20 different occurrences  $\binom{6}{3}$  can be expanded to three different scenarios:  $\{\{i, j\}, \{i, k\}, \{i, l\}\}$ ,  $\{\{i, j\}, \{k, l\}, \{i, k\}\}$  and  $\{\{i, j\}, \{j, k\}, \{i, k\}\}$ . Again, lemma 5 prevents the second scenario  $\{\{i, j\}, \{k, l\}, \{i, k\}\}$  from occurring. The transformations of the first and the third cases are in table 5 and 6. Example transformations are shown in figure 5.  $\square$

**Theorem 3.** Any function  $h(\mathbf{x}, z_1, z_2, \dots, z_k)$  in  $\mathcal{F}^2$  that is linear in  $\mathbf{z}$  can be transformed to some function  $h'(\mathbf{x}, z_f, z_b)$  in  $\mathcal{F}^2$  where  $z_f$  and  $z_b$  correspond to the forward and backward reference partitions respectively.

*Proof.* Every  $z_i$  is independent of every other  $z_j$  due to the absence of bilinear terms  $z_i z_j$ . Hence, the minimization under  $\mathbf{z}$  can be carried out in any order.

$$\min_{z_i, z_j} h(\mathbf{x}, z_i, z_j) = \min_{z_i} \min_{z_j} h(\mathbf{x}, z_i, z_j) = \min_{z_j} \min_{z_i} h(\mathbf{x}, z_i, z_j) \quad (56)$$

Applying lemma 3, followed by theorem 2, for every AV, the function  $h(\mathbf{x}, z_1, z_2, z_3, \dots, z_k)$  can be transformed into  $\hat{h}(\mathbf{x}, \hat{z}_1, \hat{z}'_1, \dots, \hat{z}_k, \hat{z}'_k)$  where  $\hat{z}_i$  and  $\hat{z}'_i$  correspond to the forward and backward reference partitions respectively. In other words, every  $z_i$  in the original function is replaced by  $\hat{z}_i$  and  $\hat{z}'_i$ . Note that one reference partition may be sufficient in some cases. Finally we use lemma 11 to obtain  $h'(x_1, x_2, x_3, x_4, z_f, z_b)$  from  $\hat{h}$ .  $\square$

## 5.2 Interacting AVs

The earlier theorem shows the transformation when the original function  $h$  has no bilinear terms  $z_i z_j$ . The problem becomes more intricate in the presence of these terms. In the earlier case, we could define partitions using a single variable. Here, it is necessary to consider the partitions using two or more variables. Below, we show the joint partition that can solve the transformation with interactions between the AVs. We refer to this as the *matroidal generators*, since the associated partitions satisfy matroid constraints (See appendix).

**Definition 10.** *The matroidal generators associated with two AVs  $z_{j1}$  and  $z_{j2}$  for expressing all graph-representable fourth order functions is given below:*

$$B \in \mathcal{B}_{j1} \iff |B| \geq 3, \mathcal{A}_{j1} = \mathcal{P} \setminus \mathcal{B}_{j1} \quad (57)$$

$$B \in \mathcal{B}_{j2} \iff |B| \geq 2, \mathcal{A}_{j2} = \mathcal{P} \setminus \mathcal{B}_{j2} \quad (58)$$

In Figure 3 we show the matroidal generators for fourth order functions. These partitions are same as the reference partitions studied earlier. The expressive power of these AVs are enhanced by interaction or the usage of the bilinear term  $z_{j1} z_{j2}$ .

**Theorem 4.** *Any function  $h(\mathbf{x}, z_1, z_2, \dots, z_k)$  in  $\mathcal{F}^2$  that has bilinear terms  $z_i z_j$  can be transformed to some function  $h'(\mathbf{x}, z_{j1}, z_{j1})$  in  $\mathcal{F}^2$ .*

*Proof.* The basic idea of the proof is to decompose a given fourth order function using the result of [23] and show that all the spawned MBFs can be expressed by the matroidal generators. Using Theorem 5.2 from [23] we can decompose a given submodular function in  $\mathcal{F}^4$  into 10 different groups  $\mathcal{G}_i, i = \{1..10\}$  where each  $\mathcal{G}_i$  is in Table 1.

Each group  $\mathcal{G}_i$  contains three or four functions giving rise to a total of 30 or more different functions. Prior work uses one auxiliary variable for every function, whereas we will show that the two AVs corresponding to the matroidal generators are sufficient to simultaneously model all these functions. As shown in [31] the functions in  $\mathcal{G}_{10}$  are not graph-representable. Note that the functions in  $\mathcal{G}_{10}$  does not become graph-representable when combined with other generators of  $\mathcal{F}^4$  according to Theorem 16(3) in [31]. We also observe that these functions are not representable by both non-interacting and interacting AVs. Thus the largest subclass  $\mathcal{F}_2^k$  should be composed of functions in the remaining 9 groups.

As the functions present in the groups  $\mathcal{G}_i, i = \{1..8\}$  do not require bilinear AV terms, any sum of functions in  $\mathcal{G}_i, i = \{1..8\}$  can be expressed with only two AVs  $z_f$  and  $z_b$  according to Theorem 3. We consider the functions in  $\mathcal{G}_9$ . The sum of functions in this group may lead to two alternatives. The union of functions in  $\mathcal{G}_9$  may either result in a function in  $\mathcal{G}_9$  or a function that uses the AVs  $z_f$  and  $z_b$ . Any function

Group	$f(\mathbf{x})$	$\min_{z_1, z_2} h(\mathbf{x}, z_1, z_2)$ where $h(\mathbf{x}, z_1, z_2) \in \mathcal{F}^2$
$\mathcal{G}_1$	$-x_i x_j$	$-x_i x_j$
$\mathcal{G}_2$	$-x_i x_j x_k$	$\min_z (2 - x_i - x_j - x_k)$
$\mathcal{G}_3$	$-x_1 x_2 x_3 x_4$	$\min_z (3 - x_1 - x_2 - x_3 - x_4)$
$\mathcal{G}_4$	$-x_1 x_2 x_3 x_4 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_3 x_4 +$ $x_2 x_3 x_4 - x_1 x_2 - x_1 x_3 - x_1 x_4 -$ $x_2 x_3 - x_2 x_4 - x_3 x_4$	$\min_z (z(1 - x_1 - x_2 - x_3 - x_4))$
$\mathcal{G}_5$	$x_i x_j x_k x_l - x_i x_j x_k - x_i x_l - x_j x_l -$ $x_k x_l$	$\min_z (z(2 - x_i - x_j - x_k - 2x_l))$
$\mathcal{G}_6$	$x_i x_j x_k - x_i x_j - x_i x_k - x_j x_k$	$\min_z (z(1 - x_i - x_j - x_k))$
$\mathcal{G}_7$	$x_i x_j x_k x_l - x_i x_j x_k - x_i x_j x_l - x_i x_k x_l$	$\min_z (z(3 - 2x_i - x_j - x_k - x_l))$
$\mathcal{G}_8$	$2x_1 x_2 x_3 x_4 - x_1 x_2 x_3 - x_1 x_2 x_4 - x_1 x_3 x_4 -$ $x_2 x_3 x_4$	$\min_z (z(2 - x_1 - x_2 - x_3 - x_4))$
$\mathcal{G}_9$	$x_i x_j x_k x_l - x_i x_j - x_i x_k - x_i x_k x_l - x_j x_k x_l$	$\min_{z_1, z_2} (z_1 + 2z_2 - z_1 z_2 -$ $z_1 x_i - z_1 x_j - z_2 x_k - z_2 x_l)$
$\mathcal{G}_{10}$	$-x_i x_j x_k x_l + x_i x_k x_l + x_j x_k x_l -$ $x_i x_k - x_i x_l - x_j x_k - x_j x_l - x_k x_l$	$f(\mathbf{x}) \ni \mathcal{F}_2^4$ as shown in [31]

**Table 1.** The above table is adapted from Figure 2 of [32] where  $\{i, j, k, l\} = S_4$ . Each group has several terms depending on the values of  $\{i, j, k, l\}$ . As the groups  $\mathcal{G}_4$  and  $\mathcal{G}_8$  are symmetric with respect to  $\{i, j, k, l\}$ ; they contain one function each.

in  $\mathcal{G}_9$  can be expressed using two AVs  $z_{91}$  and  $z_{92}$  [30]. As a result, the sum of functions in  $\mathcal{G}_i, i = \{1..9\}$  can be expressed using four AVs ( $z_f, z_b, z_{91}, z_{92}$ ). These four AVs could be merged into two AVs  $z_{j1}$  and  $z_{j2}$  in the matroidal generators as shown in Figure 3.

Hence, all functions in  $\mathcal{G}_i, i = \{1..9\}$  can be expressed by the matroidal generators.  $\square$

## 6 Linear Programming solution

For a given function  $f(x_1, x_2, x_3, x_4)$  in  $\mathcal{F}_s^4$ , our goal is to compute a function  $h(\mathbf{x}, \mathbf{z})$  in  $\mathcal{F}^2$ . As a result of theorem 4 we only need to solve the case with two AVs ( $z_{j1}, z_{j2}$ ) associated with the matroidal generators. The required function  $h(\mathbf{x}, \mathbf{z})$  is:

$$h(\mathbf{x}, z_{j1}, z_{j2}) = b_0 + \sum_i b_i x_i - \sum_{i>j} b_{ij} x_i x_j - (g_{j1} - \sum_{i=1}^4 g_{j1,i} x_i) z_{j1} + (g_{j2} - \sum_{i=1}^4 g_{j2,i} x_i) z_{j2} - j_{12} z_{j1} z_{j2}. \quad (59)$$

such that  $b_{ij}, g_{j1,i}, g_{j2,i}, j_{12} \geq 0$  and  $i, j \in S_4$ . As we know the partition of  $(z_{j1}, z_{j2})$  we know their Boolean values for all labellings of  $\mathbf{x}$ . We need the coefficients  $(b_i, b_{ij}, j_{12}, g_{j1}, g_{j2}, g_{j1,i}, g_{j2,i}), i = S_4$  to compute  $h(x_1, x_2, x_3, x_4, z_{j1}, z_{j2})$ . These coefficients satisfy both submodularity constraints (that the coefficients of all bilinear terms  $(x_i x_j, x_i z_{j1}, x_j z_{j2}, z_{j1} z_{j2})$  are less than or equal to zero) and those imposed by the reference partitions. First we list these conditions below:

$$\underbrace{\begin{pmatrix} b_{ij} \\ g_{j1,i} \\ g_{j2,i} \\ j_{12} \end{pmatrix}}_{S_p}^T \geq \mathbf{0}, i, j = S_4, i \neq j \quad (60)$$

where  $\mathbf{0}$  refers of a vector composed 0's of appropriate length. Next we list the conditions which guarantee  $f(\mathbf{x}) = \min_{z_{j1}, z_{j2}} h(\mathbf{x}, z_{j1}, z_{j2})$  for all  $\mathbf{x}$ . Let  $\forall S \in \mathcal{P}$ , and let the value of  $z_{j1}z_{j2}$  for different subsets  $S$  be given by  $\eta(S)$ . As we know the partition functions of both  $z_{j1}$  and  $z_{j2}$  it is easy to find this. Let  $\mathcal{G}$  and  $\mathcal{H}$  denote values of  $f$  and  $h$  for different  $S$ :

$$\mathcal{G} = f(\mathbf{1}_1^S, \mathbf{1}_2^S, \mathbf{1}_3^S, \mathbf{1}_4^S) \quad (61)$$

$$\mathcal{H} = h(\mathbf{1}_1^S, \mathbf{1}_2^S, \mathbf{1}_3^S, \mathbf{1}_4^S, 0, 0) - (g_{j1} - \sum_{i=1}^4 g_{j1,i} \mathbf{1}_i^S) - (g_{j2} - \sum_{i=1}^4 g_{j2,i} \mathbf{1}_i^S) - j_{12} \eta(S) \quad (62)$$

As a result we have the following 16 linear equations (N.B. there are  $2^4(16)$  different  $S$ ):

$$\mathcal{G} = \mathcal{H}, \forall S \in \mathcal{P} \quad (63)$$

Note that as with section 5 we do not make use of either auxiliary variables or the min operator over  $\mathcal{H}$ . Again, this because we already know the partition of  $(z_{j1}, z_{j2})$  and their appropriate values a priori. This can be seen as (63) need not hold if  $z_{j1}$  and  $z_{j2}$  do not lie in the reference partitions.

$$\underbrace{\begin{pmatrix} g_f - \sum_{i=1}^4 g_{f,i} \mathbf{1}_i^S \\ g_b - \sum_{i=1}^4 g_{b,i} \mathbf{1}_i^D \end{pmatrix}}_{\mathcal{G}_g} \geq \mathbf{0}, S \in \mathcal{A}_{j1}, D \in \mathcal{A}_{j2}$$

$$\underbrace{\begin{pmatrix} g_f - \sum_{i=1}^4 g_{f,i} \mathbf{1}_i^S \\ g_b - \sum_{i=1}^4 g_{b,i} \mathbf{1}_i^D \end{pmatrix}}_{\mathcal{G}_l} \leq \mathbf{0}, S \in \mathcal{B}_{j1}, D \in \mathcal{B}_{j2}.$$

Essentially we need to compute the coefficients  $(b_{ij}, g_{j1}, g_{j1,i}, g_{j2}, g_{j2,i}, j_{12})$  that satisfy the equations (60,63,64) This is equivalent to finding a feasible point in a linear programming problem:

$$\min \text{const} \quad (64)$$

$$s.t \mathcal{S}_p \geq \mathbf{0}, \mathcal{G} = \mathcal{H}, \mathcal{G}_g \geq \mathbf{0}, \mathcal{G}_l \leq \mathbf{0} \quad (65)$$

As discussed in section 4, by using a different cost function we can formulate a problem to to compute a function in  $\mathcal{F}^2$  closest to a given arbitrary fourth-order function.

## 7 Discussion and open problems

We observe that the basis MBFs corresponding to reference partitions always satisfy matroid constraints (See appendix). It can be easily shown that for  $k = 3$  there is only one reference partition corresponding to a uniform matroid  $\mathcal{U}_1$ . When  $k = 4$  we have two reference partitions corresponding to uniform matroids  $\mathcal{U}_1$  and  $\mathcal{U}_2$ . Thus we conjecture that we can transform a large subclass, possibly the largest, of  $\mathcal{F}_2^k$  using  $k - 2$  matroidal generators. Each of these generators correspond to uniform matroids  $\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \dots, \mathcal{U}_{k-2}$ . We do not have any proof for this result. However, our intuition is based on the following reasons:

- The reference partitions for  $k = 3$  and  $k = 4$  are symmetrical with respect all  $x_i$  variables.
- The reference partitions correspond to only distinct uniform matroids.
- We can only transform a subclass of all submodular functions of order  $k$ . Using the result of Zivny et al., we know that when  $k \geq 4$ , not all submodular functions can be transformed to a quadratic PBF.
- Although we use only a linear number of auxiliary variables, the underlying function is powerful as we employ all possible interactions among the auxiliary variables. Each of these intersection can be seen as the intersection of two uniform matroids.

## References

1. A. Billionnet and M. Minoux. Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for supermodular cubic functions. *Discrete Appl. Math.*, 1985.
2. E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Appl. Math.*, 123(1-3):155–225, 2002.
3. W.H. Cunningham. On submodular function minimization. *Combinatorica*, 1985.
4. J. Edmonds. Submodular functions, matroids and certain polyhedra. *Calgary International Conference on Combinatorial Structures and their applications*, 1969.
5. L. Fleischer and S. Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 2001.
6. G. Gallo and B. Simeone. On the supermodular knapsack problem. *Mathematical Programming: Series A and B*, 45(2):295–309, 1989.
7. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1981.
8. P. L. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13:388–399, 1965.
9. H. Ishikawa. Exact optimization for Markov random fields with convex priors. *PAMI*, 25:1333–1336, 2003.
10. H. Ishikawa. Higher-order clique reduction in binary graph cut. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
11. S. Iwata. A fully combinatorial algorithm for submodular function minimization. *J. Comb. Theory Ser. B*, 2000.
12. S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 2001.
13. Satoru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM J. Computing*, 2003.
14. D. Kleitman. On dedekind’s problem: The number of boolean functions. *Amer. Math Society*, 1969.
15. P. Kohli, M. P. Kumar, and P. H. S. Torr.  $P^3$  & beyond: Solving energies with higher order cliques. In *CVPR*, 2007.
16. V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2), 2004.
17. A.D. Korshunov. The number of monotone boolean functions. *Problemy Kibernet* 38:5-108, 1981.
18. Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip Torr. Graph cut based inference with co-occurrence statistics. In *European Conference on Computer Vision*. springer, 2010.
19. X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order Markov random fields. In *ECCV*, pages 269–282, 2006.
20. L. Lovasz. Submodular functions and convexity. *Mathematical Programming - The State of the Art*, 1983.
21. Mukund Narasimhan and Jeff A. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Uncertainty in Artificial Intelligence*, pages 404–412, 2005.
22. J.B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 2009.
23. S. Promislow and V. Young. Supermodular functions on finite lattices. *Order* 22(4), 2005.
24. M. Queyranne. Minimizing symmetric submodular functions. *SODA*, 1995.
25. S. Ramalingam, P. Kohli, K. Alahari, and P.H.S. Torr. Exact inference in multi-label crfs with higher order cliques. In *CVPR*, 2008.
26. D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology, 2006.
27. A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory*, 1999.
28. Alan Yuille, Anand Rangarajan, and A. L. Yuille. The concave-convex procedure (cccp). In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.
29. B. Zalesky. Efficient determination of gibbs estimators with submodular energy functions. <http://arxiv.org/abs/math/0304041v1>, 2003.
30. S. Zivny, D.A.Cohen, and P.G. Jeavons. The expressive power of binary submodular functions. *Discrete Applied Mathematics*, 2009.
31. S. Zivny and P.G. Jeavons. Classes of submodular constraints expressible by graph cuts. *Proceedings of CP*, 2008.
32. S. Zivny and P.G. Jeavons. Which submodular functions are expressible using binary submodular functions? *Oxford University Computing Laboratory Research Report CS-RR-08-08*, 2008.

## A Tables

i	$\min(0, \kappa(s, \{i, j, k\}))$	$\min(0, \kappa(s, \{i, j, l\}))$	$\min(0, \kappa(s, \{i, j, k, l\}))$	$\kappa(f, \{i, j\})$
1	0	0	0	0
2	0	$\kappa(s, \{i, j, l\})$	$\kappa(s, S_4)$	$g_{s,k}$
3	$\kappa(s, \{i, j, k\})$	0	$\kappa(s, \{i, j, k, l\})$	$g_{s,l}$
4	$\kappa(s, \{i, j, k\})$	$\kappa(s, \{i, j, l\})$	$\kappa(s, \{i, j, k, l\})$	$\kappa(s, \{i, j\})$

**Table 2.** See lemma 4. In all four cases  $\kappa(f, \{i, j\})$  is non-negative. This result holds for the fourth case as  $\kappa(s, \{i, j\}) \geq 0$ .

Case 1: $\{i, j\} \in \mathcal{B}_s$ .		
$h_2 = \kappa(s, \{i, j\})x_i x_j + \underbrace{(2 * g_s - g_{s,i} - g_{t,j})}_{g_t} - \underbrace{(g_s - g_{s,i})}_{g_{t,j}} x_i - \underbrace{(g_s - g_{s,j})}_{g_{t,i}} x_j -$ $\underbrace{g_{s,k} x_k}_{g_{t,k}} - \underbrace{g_{s,l} x_l}_{g_{t,l}} z_t$		
$S$	$\kappa(t, S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i, j\}$	0	$S \in \mathcal{A}_t$
$\{i, k\}$	$\kappa(s, \{j, k\})$	$S \in \mathcal{A}_t$ since $\{j, k\} \in \mathcal{A}_s$
$\{k, l\}$	$\kappa(s, \{i, k\}) + \kappa(s, \{j, l\})$	$S \in \mathcal{A}_t$ since $\{i, k\}, \{j, l\} \in \mathcal{A}_s$
$\{i, j, k\}$	$-g_{s,k}$	$S \in \mathcal{B}_t$
$\{i, k, l\}$	$\kappa(s, \{j, k, l\})$	$S \in \mathcal{B}_t$ since $\{j, k, l\} \in \mathcal{B}_s$

**Table 3.** See theorem 2. Case 1: The details of the transformation (similar to one in equation (55)) are shown for a scenario where exactly one set  $\{i, j\}$  with cardinality two exist in  $\mathcal{B}_s$ . We prove that after the transformation all the sets  $S$  with  $|S| = 2$  exist in  $\mathcal{A}_t$  and  $|S| \geq 3$  exist in  $\mathcal{B}_t$ . Although the reduction is illustrated for only a few cases, they are representative of the remainder.

## B Definitions

**Definition 11.** A matroid  $\mathcal{M}$  is an ordered pair  $(E, \mathcal{I})$  consisting on a finite set  $E$  and a family of subsets  $\mathcal{I}$  of  $E$  satisfying the following conditions:

1.  $\emptyset \in \mathcal{I}$ .
2. If  $I \in \mathcal{I}$  and  $I' \subseteq I$ , then  $I' \in \mathcal{I}$ .
3. If  $I_1$  and  $I_2$  are in  $\mathcal{I}$  and  $|I_1| < |I_2|$ , then there is an element  $e$  of  $I_2 - I_1$  such that  $I_1 \cup e \in \mathcal{I}$ .

The maximal independent set in a matroid is called the base of a matroid. All the bases of a matroid are equicardinal, i.e., they have the same number of elements.

**Definition 12.** The dual matroid of  $\mathcal{M}$  is given by  $\mathcal{M}^*$  whose bases are the complements of the bases of  $\mathcal{M}$ .

**Definition 13.** In a uniform matroid  $\mathcal{U}_n(E, \mathcal{I})$ , all the independent sets  $I_i \in \mathcal{I}$  satisfy the condition that  $|I_i| \leq n$  for some fixed  $n$ .

Case 2: $\{i, j\}, \{j, k\} \in \mathcal{B}_s$ .		
$h_2 = \kappa(s, \{i, j\})x_i x_j + \kappa(s, \{j, k\})x_j x_k + \underbrace{(3g_s - 2g_{s,j} - g_{s,i} - g_{s,k} -}_{g_t} \underbrace{(g_s - g_{s,j})x_i - (2g_s - g_{s,i} - g_{s,j} - g_{s,k})x_j - (g_s - g_{s,j})x_k - (g_{s,l}x_l)z_t}_{g_{t,i} \quad g_{t,j} \quad g_{t,k} \quad g_{t,l}}$		
$S$	$\kappa(t, S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i, j\}$	0	$S \in \mathcal{A}_t$
$\{i, l\}$	$\kappa(s, \{i, k\}) + \kappa(s, \{j, l\})$	$S \in \mathcal{A}_t$ since $\{i, k\}, \{j, l\} \in \mathcal{A}_s$
$\{j, l\}$	$\kappa(s, \{j\}) + g_{s,l}$	$S \in \mathcal{A}_t$ since $\{j\} \in \mathcal{A}_s$ and $g_{s,l} \geq 0$
$\{i, j, k\}$	$-\kappa(s, \{j\})$	$S \in \mathcal{B}_t$ since $\{j\} \in \mathcal{A}_s$
$\{i, k, l\}$	$\kappa(s, \{i, k, l\})$	$S \in \mathcal{B}_t$ if $\{i, k, l\} \in \mathcal{B}_s$
$\{i, j, l\}$	$-g_{s,l}$	$S \in \mathcal{B}_t$

**Table 4. See theorem 2.** Case 2: We study the scenario where exactly two sets with cardinality two  $\{\{i, j\}, \{j, k\}\}$  occur in  $\mathcal{B}_s$ . Note that all other cases either can not happen (according to lemma 5) or similar to the ones shown in this table. We also prove that after the transformation all the sets  $S$  with  $|S| = 2$  exist in  $\mathcal{A}_t$  and  $|S| \geq 3$  exist in  $\mathcal{B}_t$ .

Case 3: $\{i, j\}, \{i, k\}, \{i, l\} \in \mathcal{B}_s$ .		
$h_2 = \kappa(s, \{i, j\})x_i x_j + \kappa(s, \{i, k\})x_i x_k + \kappa(s, \{i, l\})x_i x_l + \underbrace{(\min(0, \kappa(s, \{j, k, l\})) + 3(g_s - g_{s,i}) - \min(0, \kappa(s, \{j, k, l\}))) + 2(g_s - g_{s,i})}_{g_t} x_i - \underbrace{(g_s - g_{s,i})x_j - (g_s - g_{s,i})x_k - (g_s - g_{s,i})x_l}_{g_{t,j} \quad g_{t,k} \quad g_{t,l}} z_t$		
$S$	$\kappa(t, S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i, j\}$	0	$S \in \mathcal{A}_t$
$\{j, k\}$	$\min(0, \kappa(s, \{j, k, l\})) + (g_s - g_{s,i})$	$S \in \mathcal{A}_t$ since $\{i\} \in \mathcal{A}_s$
$\{i, j, k\}$	$-\kappa(s, \{j\})$	$S \in \mathcal{B}_t$ since $\{j\} \in \mathcal{A}_s$
$\{i, k, l\}$	$\kappa(s, \{i, k, l\})$	$S \in \mathcal{B}_t$ if $\{i, k, l\} \in \mathcal{B}_s$
$\{i, j, l\}$	$-g_{s,l}$	$S \in \mathcal{B}_t$

**Table 5. See theorem 2.** Case 3: Here we study the scenario where exactly three sets with cardinality two  $\{\{i, j\}, \{i, k\}, \{i, l\}\}$  exist in  $\mathcal{B}_s$ . The only other case where three sets can exist is shown in table 6. The shown cases are generalizations of all the possible cases that can occur without violating lemma (5). We prove that after the transformation all the sets  $S$  with  $|S| = 2$  exist in  $\mathcal{A}_t$  and  $|S| \geq 3$  exist in  $\mathcal{B}_t$ .

Case 4: $\{i, j\}, \{i, k\}, \{i, l\} \in \mathcal{B}_s$ .		
$h_2 = \kappa(s, \{i, j\})(1 - x_i - x_j - x_k) - (g_{s,k} - g_{s,j})x_i x_k - (g_{s,k} - g_{s,i})x_j x_k +$ $\underbrace{2(g_s - g_{s,k})}_{g_t} - \underbrace{(g_s - g_{s,k})}_{g_{t,i}} x_i - \underbrace{(g_s - g_{s,k})}_{g_{t,j}} x_j - \underbrace{(g_s - g_{s,k})}_{g_{t,k}} x_k - \underbrace{g_{s,l}}_{g_{t,l}} x_l) z_t +$ $\underbrace{(-2\kappa(s, \{i, j\}))}_{g_r} - \underbrace{(-\kappa(s, \{i, j\}))}_{g_{r,i}} (1 - x_i) - \underbrace{(-\kappa(s, \{i, j\}))}_{g_{r,j}} (1 - x_j) -$ $\underbrace{(-\kappa(s, \{i, j\}))}_{g_{r,k}} (1 - x_k) - \underbrace{0}_{g_{r,l}} (1 - x_l) z_r$		
$S$	$\kappa(t, S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i, j\}$	0	$S \in \mathcal{A}_t \kappa = 0$
$\{i, l\}$	$\kappa(s, \{k, l\})$	$S \in \mathcal{A}_t$ since $\{k, l\} \in \mathcal{A}_s$
$\{i, j, k\}$	$-\kappa(s, \{k\})$	$S \in \mathcal{B}_t$ since $\{k\} \in \mathcal{A}_s$
$\{i, j, l\}$	$-g_{s,l}$	$S \in \mathcal{B}_t$ since $g_{s,l} \geq 0$
$S$	$\kappa(r, S)$	$S \in \mathcal{A}_r$ or $S \in \mathcal{B}_r$
$\{i, l\}$	0	$S \in \mathcal{A}_r$
$\{i, j\}$	$-\kappa(s, \{i, j\})$	$S \in \mathcal{A}_r$ since $\{i, j\} \in \mathcal{B}_s$
$\{i\}$	0	$S \in \mathcal{B}_r$
$\{l\}$	$\kappa(s, \{i, j\})$	$S \in \mathcal{B}_r$ since $\{i, j\} \in \mathcal{B}_s$

**Table 6.** See theorem 2. Case 4: We consider three sets  $\{i, j\}, \{i, k\}, \{j, k\} \in \mathcal{B}_s$  which involve only three elements and all three repeating in more than one set. Without loss of generality, we assume that  $\kappa(s, \{i, j\}) \geq \kappa(s, \{i, k\})$  and  $\kappa(s, \{i, j\}) \geq \kappa(s, \{j, k\})$ . In this case we replace the AV  $z_s$  using two variables  $z_t$  and  $z_r$ .

## C Useful Lemmas

It is not completely clear as to why a few basis AVs can replace several hundreds of AVs in a function in  $\mathcal{F}_2^4$ . We observed some differences in the general partitions and reference partitions. We found that not all partitions satisfy the conditions of a matroid. However, the reference partitions form matroids in third and fourth order functions. We summarize these results in the following two lemmas.

**Lemma 6.** *The ordered pair  $\{S_4, \mathcal{A}\}$  corresponding to all partitions do not form a matroid.*

*Proof.* An ordered pair  $(E, \mathcal{I})$  is a matroid if it satisfies the three conditions given in Definition 11. The first condition is to show that  $\emptyset \in \mathcal{I}$ . As per the definition of the partition,  $\{\emptyset, \mathcal{P}\}$  is a valid partition where  $\mathcal{A} = \emptyset$ . The second matroid condition can be obtained by using lemma 9 in the reverse direction for subsets of  $\mathcal{A}$ . The third matroid conditions states that if  $|I_1| < |I_2|$  and  $I_1, I_2 \in \mathcal{I}$  then there exists an element  $e$  in  $I_2$  such that  $I_1 \cup e \in \mathcal{I}$ . However, this condition is not true for all partitions. For example, consider a partition  $\{\mathcal{A} = \{\{\emptyset\}, \{1\}, \{2\}, \{3\}, \{4\}, \{3, 4\}\}, \mathcal{B} = \mathcal{P}/\mathcal{A}\}$ . Let  $I_1 = \{1\}$  and  $I_2 = \{3, 4\}$  be the two independent sets. Although  $|I_1| < |I_2|$ , there is no element  $e$  in  $I_2$  satisfying  $I_1 \cup e \in \mathcal{A}$ .  $\square$

**Lemma 7.** *The ordered pair  $\{S_4, \mathcal{A}_f\}$  corresponding to the forward reference partition is a uniform matroid  $\mathcal{U}_2$  (See Definition 13).*

*Proof.* It can be easily seen that the subsets of  $\mathcal{A}_f$  satisfy the three conditions given in Definition 11. In addition, every  $A \in \mathcal{A}$  satisfies the condition  $|A| \leq 2$ . Thus  $\{S_4, \mathcal{A}_f\}$  forms a uniform matroid  $\mathcal{U}_2$ .  $\square$

**Lemma 8.** *The ordered pair  $\{S_4, \mathcal{B}_b\}$  corresponding to the forward reference partition is a uniform matroid  $\mathcal{U}_1$  (See Definition 13).*



*Proof.* It can be easily seen that the subsets of  $\mathcal{B}_b$  satisfy the three conditions given in Definition 11. In addition, every  $B \in \mathcal{B}_b$  satisfies the condition  $|B| \leq 2$ . Thus  $\{S_4, \mathcal{B}_b\}$  forms a uniform matroid  $\mathcal{U}_1$ .  $\square$

It can also be shown that the ordered pair  $\{S_4, \mathcal{B}_f\}$  is the dual of a uniform matroid  $\{S_4, \mathcal{A}_f\}$  (See Definition 12). Similarly the ordered pair  $\{S_4, \mathcal{A}_b\}$  is the dual of a uniform matroid  $\{S_4, \mathcal{B}_b\}$ .

The partitions are nothing but lattices and thus they satisfy the following property.

**Lemma 9.** *Every AV  $z_s$  that is associated with a partition separates  $\mathcal{P}$  into sets  $\mathcal{A}_s$  and  $\mathcal{B}_s$  such that if any set  $B \in \mathcal{B}_s$ , then every set  $S \supseteq B$  is also an element of  $\mathcal{B}_s$ .*

*Proof.* If set  $B \in \mathcal{B}_s$  then  $\kappa(s, B) = (g_s - \sum_{i=1}^4 g_{s,i} \mathbf{1}_i^B) \leq 0$ . Since  $S \supseteq B$  and  $g_{s,i} \geq 0, \forall i = S_4$  we have  $(g_s - \sum_{i=1}^4 g_{s,i} \mathbf{1}_i^S) \leq (g_s - \sum_{i=1}^4 g_{s,i} \mathbf{1}_i^B)$ . This implies that the partition coefficient  $\kappa(s, S) \leq 0$  and thus  $S \in \mathcal{B}_s$ .  $\square$

**Lemma 10.** *For an AV  $z_s : [\mathcal{A}_s, \mathcal{B}_s]$  if  $\mathcal{A}_s = \emptyset$  we can transform a function  $h(\mathbf{x}, z_s)$  in  $\mathcal{F}^2$  to some function  $h'(\mathbf{x})$  in  $\mathcal{F}^2$ . Similarly for an AV  $z_t : [\mathcal{A}_t, \mathcal{B}_t]$ , if  $\mathcal{B}_t = \emptyset$  we can transform a function  $h(\mathbf{x}, z_t)$  in  $\mathcal{F}^2$  to some function  $h'(\mathbf{x})$  in  $\mathcal{F}^2$ .*

*Proof.* If  $\mathcal{A}_s = \emptyset$  then  $\arg \min_{z_s} h(\mathbf{x}, z_s) = 1, \forall \mathbf{x}$ . Hence  $\min_{z_s} h(\mathbf{x}, z_s) = h(\mathbf{x}, 1) = h'(\mathbf{x})$ . Similarly when  $\mathcal{B}_t = \emptyset$ ,  $\min_{z_t} h(\mathbf{x}, z_t) = h(\mathbf{x}, 0) = h'(\mathbf{x})$ .

**Lemma 11.** *If  $z_s$  and  $z_t$  are two AVs in a function  $h(\mathbf{x}, z_s, z_t)$  in  $\mathcal{F}^2$  sharing the same partition, then  $h$  can be transformed to some  $h'(\mathbf{x}, z)$  in  $\mathcal{F}^2$  having a single AV with the same partition.*

*Proof.* The partition of  $z_s$  is independent of  $z_t$  and vice versa, since there is no  $z_s z_t$  term in  $h(\mathbf{x}, z_s, z_t)$ . Thus while studying the partition of  $z_s$  we can treat  $z_t$  as a constant. Since  $z_s$  and  $z_t$  have the same partition property,  $z_s = z_t$  at  $\arg \min_{z_s, z_t} h(\mathbf{x}, z_s, z_t), \forall \mathbf{x}$ . Thus we can replace  $z_s$  and  $z_t$  using a single variable  $z$  in an equivalent function  $h'(\mathbf{x}, z)$ .